

SMPTE REGISTERED DISCLOSURE DOCUMENT

Dolby Atmos[®] Bitstream Specification



Page 1 of 20

The attached document is a Registered Disclosure Document prepared by the proponent identified below. It has been examined by the appropriate SMPTE Technology Committee and is believed to contain adequate information to satisfy the objectives defined in the Scope, and to be technically consistent.

This document is NOT a Standard, Recommended Practice or Engineering Guideline, and does NOT imply a finding or representation of the Society.

Errors in this document should be reported to the proponent identified below, with a copy to eng@smpte.org.

This document is intended to allow the interpretation of Dolby Atmos Bitstream files. It is not intended to support the development of hardware or software applications that create or process these files. Creation and processing of such files is reserved to individuals and organizations that have entered into agreements with the proponent identified below for this purpose. Use of this document to produce or process Dolby Atmos files using non-Dolby tools would potentially cause user confusion, diminished sound quality as experienced by content consumers, and damage to the reputation of the Dolby Atmos brand and to Dolby Laboratories itself.

All other inquiries in respect of this document, including inquiries as to intellectual property requirements that may be attached to use of the disclosed technology, should be addressed to the proponent identified below.

Proponent contact information:

Andrew Poulain
Dolby Laboratories Inc.
1275 Market St.
San Francisco, CA 94103

Email andrew.poulain@dolby.com

Table of Contents		Page
1	Scope	4
2	Bitstream Organization	4
2.1	ATMOSFrame Element	5
2.2	BedDefinition Element	5
2.3	ObjectDefinition Element	5
2.4	AudioDataDLC Element	5
3	Bitstream Conventions	5
3.1	Position.....	5
3.2	Relative distance coding	6
3.3	Amplitude Gain	6
3.4	Plex Coding	6
4	Bit Stream Syntax	7
4.1	Syntax of ReadElement()	7
4.2	Syntax of ATMOSFrame()	8
4.3	Syntax of BedDefinition1().....	8
4.4	Syntax of ObjectDefinition1()	9
4.5	Syntax of AudioDataDLC().....	10
5	Bit Stream Field Description	12
5.1	ReadElement() Data Fields	12
5.2	ATMOSFrame() data fields.....	12
5.3	BedDefinition1() Data Fields.....	14
5.4	ObjectDefinition1() Data Fields	15
5.5	AudioDataDLC() Data Fields.....	18

Introduction

Dolby Atmos® is an advanced cinema sound format comprising an audio essence and metadata stream played through specialized renderers in the cinema.

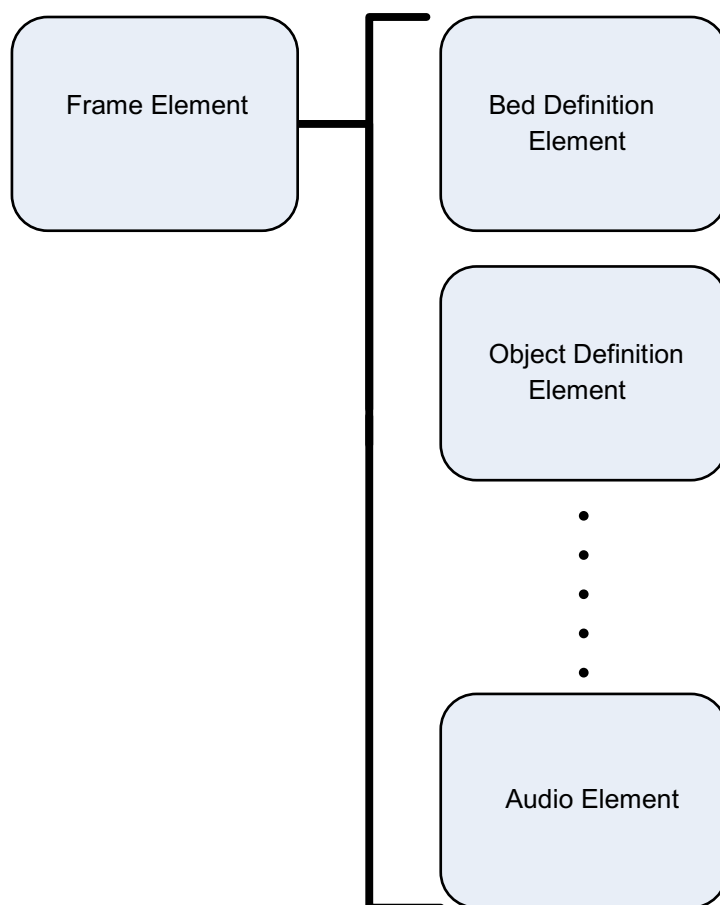
1 Scope

This document defines the syntax of a frame-based Dolby Atmos bit stream. The bit stream carries audio essence and metadata necessary to reproduce a complete audio program.

2 Bitstream Organization

The audio program is segmented into Frames, with Frames transmitted 24, 25, 30, 48, 50, 60, 96, 100, or 120 times per second. The audio Frames are aligned with the program edit units. In most cases the edit units, and picture and audio frames are of the same duration and are time aligned. Support for frame rates above 120 Hz is not defined.

All audio data is encapsulated into “elements,” similar in concept to “chunks” in the RIFF¹ format. Each element begins with a unique identifier called **ElementID**. The second field, **ElementSize**, indicates the size in bytes of the entire element, not including the **ElementID** and **ElementSize**. Elements can contain sub-Elements. The **ElementSize** includes the size of all sub elements. Sub-Elements contain additional description data related to the parent Element. At the top level, the entire audio frame is contained in a single *ATMOSFrame* element. All audio essence and metadata elements for a given frame are contained as sub Elements of the ATMOSFrame Element as shown below.



¹ Resource Interchange File Format

Currently there are 4 element types specified in the Dolby Atmos bit stream; *ATMOSFrame*, *BedDefinition*, *ObjectDefinition*, and *AudioDataDLC*. The purpose of each element is described in the following sections.

2.1 ATMOSFrame Element

The *ATMOSFrame* element contains all the information that is common to the entire Dolby Atmos frame. Specifically, the *ATMOSFrame* contains the Dolby Atmos version, audio sample rate, the audio bit depth, the audio frame rate, and the maximum number of rendered audio assets. All raw audio assets and metadata must be sub elements of the *ATMOSFrame* element.

2.2 BedDefinition Element

A Dolby Atmos bed is a collection of audio channels. An audio channel is an audio stream that is intended to be played back with a nominal location (e.g. "Left" channel) or function (e.g. LFE). The *BedDefinition* element contains a list of the audio assets and the associated channel names.

2.3 ObjectDefinition Element

The Dolby Atmos system allows audio assets to be panned to any location independent of the physical or nominal loudspeaker configuration; these panned audio assets are called objects. The *ObjectDefinition* element provides all the information to pan an audio object. Each *ObjectDefinition* element updates the position of a single audio object at approximately 20-ms time intervals.

The Dolby Atmos presentation can have a large number of audio objects that will be independently rendered to the appropriate locations. To achieve this, the Dolby Atmos bit stream will contain multiple *ObjectDefinition* elements that must be direct sub-elements of the *ATMOSFrame* element and must have a unique **MetaID**.

2.4 AudioDataDLC Element

Each *AudioDataDLC* element contains the audio assets for one track of audio, channel or object. Every audio track is losslessly compressed and exists for the duration of the program. The *AudioDataDLC* element supports sample rates of 48 kHz or 96 kHz with 24-bit resolution. All *AudioDataDLC* elements must be direct sub-elements of the *ATMOSFrame* element and must have a unique **AudioDataID**. Note: audio object tracks are typically sparse; most audio events conveyed by objects have limited time extent, with digital zero signal between events. The *AudioDataDLC* element can efficiently indicate periods of silence to dramatically decrease the audio payload.

3 Bitstream Conventions

3.1 Position

Axes and Origin

Many of the metadata elements contained in the bitstream specify a relative position or size. In most cases position is described relative to the playback environment using a unit cube to describe the room boundaries. The origin is taken to be the front left corner of the room. Position is then described using Euclidian (x,y,z) coordinates, assigned as follows:

x: lateral, or left/right position

x=0 corresponds to left wall; x=1 corresponds to right wall.

y: longitude, or front/back position

y=0 corresponds to front wall; y=1 corresponds to back wall.

z: elevation, or up/down position

z=0 corresponds to a plane aligned with the screen, side and rear loudspeakers;

z=1 corresponds to the ceiling.

For example,

(0, 0, 0) -> front left corner, 0 elevation (left screen speaker),
 (1, 0, 0) -> front right corner, 0 elevation (right screen speaker), and
 (0.5, 0.5, 1) -> middle of ceiling.

Metadata that describes position relative to the room uses the unit axes and origin described above; the location along each axis is coded using the distance coding method described below.

3.2 Relative distance coding

12 Bit Distance

Throughout the bitstream, distance metadata on or within the unit cube shall be coded as an n -bit unsigned integer (D_n) that maps linearly into the range $[0, 1]$ but the mapping depends on the axis. For the X and Y axis, the DistanceXY formula shall be used. For the Z axis, the DistanceZ formula shall be used. See section 10.1 for an explanation of the unit cube and the associated axes.

$$\text{DistanceXY} = D_n / 2^{(n-1)} - (2^{(n-1)} - 1) / 2^{(n-1)}, \quad 2^{n-1} - 1 \leq D_n \leq (2^n) - 1$$

$$\text{DistanceZ} = D_n / (2^n - 1), \quad 0 \leq D_n \leq (2^n) - 1$$

3.3 Amplitude Gain

Many of the metadata elements contained in the bitstream specify a gain or scale factor to be applied to audio data. Throughout the bitstream, amplitude information is coded as a logarithmic value using a 10 bit mantissa, A10, to express values in the range $[0, 1]$. Thus the logarithmic code word, interpreted as an unsigned integer, can be mapped to a linear value as follows:

$$\begin{aligned} \text{gain} &= 0, & A10 &= (2^{10}) - 1 \\ \text{gain} &= 2^{(-A10/64)}, & 0 &\leq A10 \leq (2^{10}) - 2 \end{aligned}$$

This provides gain factors logarithmically spaced from 0 dB to -96 dB in steps of approximately 0.094 dB, plus a gain factor of zero (-infinity dB).

3.4 Plex Coding

Element size and some other fields are coded using Plex. Plex coding allows efficient coding of small numbers, with the capability to express arbitrarily large values. The ElementSize field is Plex(8) coded to efficiently indicate elements up to 254 bytes in length, and allow for arbitrarily large Element size.

Plex(n) defines a field size of n bits that may be expanded if it is insufficient to hold the desired symbol. A data field of all ones is an escape code. If the desired symbol is not the escape code and fits within the allocated field, the symbol shall be placed therein, with leading zeros (as needed) to fill the field. Otherwise, an escape code is used to signal that a replacement symbol follows in the bitstream. The field size for the replacement symbol is twice as long as the field it replaces. This process continues, with additional escape codes and doubling of field size, until a field with size sufficient to carry the desired symbol is provided. The desired symbol shall be placed therein, with leading zeros (as needed) to fill the field. Prior to Plex coding, any leading zeros of a symbol (e.g., an unsigned integer) shall be dropped in order to fit the symbol into a smaller container. During decoding, leading zeros shall be prepended as needed to reproduce the full symbol. Plex coded symbols shall be encoded in the smallest container possible for the value to be encoded. For example, 0x1234 is to be Plex(8) encoded as 0xFF1234 instead of 0xFFFFFFFF00001234. In this document, symbols to be Plex encoded shall have a value less than or equal to 0xFFFFFFFF. Plex(8) coding is shown below.